Diese Website verwendet Cookies von Google, um Dienste anzubieten und den Traffic zu analysieren. Deine IP-Adresse und dein User-Agent werden zusammen mit Messwerten zur Leistung und Sicherheit für Google freigegeben. So können Nutzungsstatistiken generiert, Missbrauchsfälle erkannt und behoben und die Qualität des Dienstes gewährleistet werden.

WEITERE INFORMATIONEN OK

An unending variety of topics

Saturday, January 31, 2009

Secure timestamping and confidential auditing

Another interesting protocol is cryptographic timestamping. The purpose is to prove that a particular piece of content (i.e. some array of bits) existed at a particular period of time. The basic idea goes back to the anagram publication technique that Robert Hooke, Galileo, and some other early scientists used to prove that they discovered certain things long before they published them. When Hooke, for example, discovered his law of elasticity, he published the gobbledygook letters "ceiiinossttuv." Later, when he published his law of elasticity, he published "ut tensio sic vis" (as the extension, so with the force). The earlier published anagram proved that he had discovered this law long before he published it.

The modern protocol uses cryptographic hash functions instead of anagrams. Any set of bits (digital content, a network event, whatever) is passed through the hash function, turning into into a unique random-looking string of bits. Those bits are then published to multiple timestamp servers on the Internet. The timestamp servers create a chain of hashes. Using the chain of published hashes, it is easy to later prove that (1) the hash was published before one event and after another event, thus proving the time of publication, and (2) that the hash uniquely corresponds to a particular content.

Note that the content of the file does not need to be published until proof is needed. Thus, for example, one could digitally timestamp a secret digital inventor's notebook in order to prove later that the invention existed at that time. (Might be quite useful under the American first-to-invent system).

Indeed, using multiparty secure computation or the related protocols called zero-knowledge proofs, one can even make a later proof without publishing the content. For example, if Bob received a secret encrypted e-mail message from Alice, Alice and Bob could prove to the world that Alice sent a message at 18:42:39 and Bob received it at 18:43:05 without revealing the actual contents of the message. In a confidential audit of Alice's books based on securely timestamped transactions between Alice and Bob and Alice and Charles, performed using secure multiparty computation, Alice can prove to the auditor that her books balance based on real transactions with Bob and Charles, without revealing in unencrypted form to the auditor either the transactions or the books. Such is the magical reality of cryptography!

Here are some links to papers and other references on secure timestamping.

Posted by Nick Szabo at 5:31 PM

Pages

- Home

About Me

Nick Szabo

"A premier thinker about history, law and economics, and the lessons they have for security." -- Adam Shostack, Emergent Chaos

"Szabo comes out with these essays that leave me in awe." -- Brian Dunbar

"Reading material that is eclectic, challenging and endlessly fascinating." -- Sean McGrath, Propylon

"Like most blogs worth my attention, this blog is updated only infrequently. That is because the authors of blogs worth my attention only post when they have something to say that is true, relevant and not already known by their audience. Most of the human race does not have the skill to know when an idea has these three properties. The skill is particularly rare in the fields of politics and economics, which is why this blog is such a rare and valuable thing." -- Richard Hollerith

View my complete profile

4 comments:

Daniel Farina said...

One should notice this is much like Git's model where all versions are stamped with a hash that is dependent on the content as well as the lineage. recursively, it means you can chain your way through hashes unambiguously through the history until the beginning of the repository.

1:13 PM

Curtis said...

Daniel,
Only in the sense that it involves hashes. Git's model may give some witness to integrity of the repository and so on, but it doesn't have anything to do with with certifiably untampered log data.

Secure timestamping must involve Trent (a trusted, unbiased party) placing his signature on a cryptographic hash of the "stuff" he's certifying existed and the timestamp at which he "saw" and signed it.

Just because you sign a hash of a blob with a timestamp means nothing. Trent has to be trusted and he also has to publish his timestamping activity regularly (at an interval suitable for the purpose at hand).

You could easily forge a git repository with some data that says "hey this existed back in 1997" (ignoring the fact that git did not exist then) and it means nothing (other than that you are malicious, disingenuous, or don't bother updating the clock on your PC). However if you could show that Trent published a signature vouching for the existence of that data in 1997 then we might have to start pondering the true origins of git...

8:30 AM

Paul said...

Back in 2000-03, I was involved in a high-tech startup (CyberLocator Inc.) that figured out a way to time-stamp documents and other Web-transmitted material by using GPS technology in our proprietary networked methodology. It's a complex process (two patents were required to fully protect its operation) but essentially binds a time-and-location indicator to an e-mail or attachment in such a way that spoofing said data is impossible. This is a much more secure method than the one you describe. It's too bad the company went out of business before we could bring the service to market.

It's also a shame that this post has become a haven for spammers in the comments area.

9:08 AM

Anonymous said...

First of all, get the spam off your site. Second of all, due to "the way git works", given a hash from "NOW", you can prove the lineage of that hash.

Given that you can prove the lineage of any hash from "NOW", you can prove that it is ~impossible~ to insert something into that lineage (such that sha-256 remains unbroken).

Blog Archive

- ► 2018 (1)
- ► 2017 (3)
- ► 2016 (4)
- ► 2015 (3)
- ► 2014 (3)
- ► 2013 (3)
- ► 2012 (8)
- ► 2011 (12)
- ► 2010 (9)
- ▼ 2009 (29)
  - ► November (3)
  - ► October (6)
  - ► September (3)
  - ► August (1)
  - ► July (1)
  - ► June (3)
  - ► May (6)
  - ► April (3)
  - ▼ January (3)
    - Law embedded in the world
    - Confidential auditing
    - Secure timestamping and confidential auditing
- ► 2008 (55)
- ► 2007 (47)
- ► 2006 (130)
- ► 2005 (44)

So, yes you can forge:

a,b,c,XXX,x,y,z

But given the hash "f" you cannot forge:

a,b,c,XXX,d,e,f

..."f" can only come from the lineage of a,b,c,d,e,f.

True that this must all happen "in the open" and "with trusted parties", etc. But to the extent that you trust the person who gave you 'f', you trust that the lineage is actually a,b,c,d,e,f and that 'c' and 'd' must necessarily be ordered prior to 'f' in the history.

--Robert

12:58 PM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Simple theme. Powered by Blogger.